10 Rec'd PCT/PTO 13 AUG 1999

OBJECT-BASED AUDIO-VISUAL TERMINAL AND BITSTREAM STRUCTURE

Technical Field

This invention relates to the representation, transmission, processing and display of video and audio-visual information, more particularly of object-based information.

Background of the Invention

Image and video compression techniques have been developed which, unlike traditional waveform coding, attempt to capture high-level structure of visual content. Such structure is described in terms of constituent "objects" which have immediate visual relevancy, representing familiar physical objects, e.g. a ball, a table, a person, a tune or a spoken phrase. Objects are independently encoded using a compression technique that gives best quality for each object. The compressed objects are sent to a terminal along with composition information which tells the terminal where to position the objects in a scene. The terminal decodes the objects and positions them in the scene as specified by the composition information. In addition to yielding coding gains, object-based representations are beneficial with respect to modularity, reuse of content, ease of manipulation, ease of interaction with individual image components, and integration of natural, camera-captured content with synthetic, computer-generated content.

Summary of the Invention

In a preferred architecture, structure or format for information to be processed at an object-based video or audio-visual (AV) terminal, an object-oriented bitstream includes objects, composition information, and

scene demarcation information. The bitstream structure allows on-line editing, e.g. cut and paste, insertion/deletion, grouping, and special effects.

In the preferred architecture, in the interest of ease of editing, AV objects and their composition information are transmitted or accessed on separate logical channels (LCs). The architecture also makes use of "object persistence", taking advantage of some objects having a lifetime in the decoder beyond their initial presentation time, until a selected expiration time.

Brief Description of the Drawing

Fig. 1 is a functional schematic of an exemplary object-based audio-visual terminal.

Fig. 2a is a schematic of an exemplary object-based audio-visual composition packet.

Fig. 2b is a schematic of an exemplary object-based audio-visual data packet.

Fig. 2c is a schematic of an exemplary compound composition packet.

Fig. 3 is a schematic of exemplary node and scene description information using composition.

Fig. 4 is a schematic of exemplary stream-node association information.

Fig. 5 is a schematic of exemplary node/graph update information using a scene.

Fig. 6 is a schematic of an exemplary audio-visual terminal design.

Fig. 7 is a schematic of an exemplary audio-visual system controller in the terminal according to Fig. 6.

Fig. 8 is a schematic of exemplary information flow in the controller according to Fig. 7.

Detailed Description

An audio-visual (AV) terminal is a systems component which is instrumental in forming, presenting or displaying audio-visual content. This includes (but is not limited to) end-user terminals with a monitor screen and loudspeakers, as well server and mainframe computer facilities in which audio-visual information is processed. In an AV terminal, desired functionality can be hardware-, firmware- or software-implemented. Information to be processed may be furnished to the terminal from a remote information source via a telecommunications channel, or it may be retrieved from a local archive, for example. An object-oriented audio-visual terminal more specifically receives information in the form of individual objects, to be combined into scenes according to composition information supplied to the terminal.

Fig. 1 illustrates such a terminal, including a de-multiplexer (DMUX) 1 connected via a logical channel LC0 to a system controller or "executive" 2 and via logical channels LC1 through LCn to a buffer 3. The executive 2 and the buffer 3 are connected to decoders 4 which in turn are connected to a composer unit 5. Also, the executive 2 is connected to the composer unit 5 directly, and has an external input for user interaction, for example.

In the preferred AV architecture, the AV objects and their composition information are transmitted or accessed on separate logical channels. The DMUX receives the Mux2 layer from the lower layers and de-multiplexes it into logical channels. LC0 carries composition information which is passed on to the executive. The AV objects received on other logical channels are stored in the buffer to be acted upon by the

decoders. The executive receives the composition information, which includes the decoding and presentation time stamps, and instructs the decoders and composer accordingly.

The system handles object composition packets (OCP) and object data packets (ODP). A composition packet contains an object's ID, time stamps and the "composition parameters" for rendering the object. An object data packet contains an object ID, an expiration time stamp in case of persistent objects, and object data.

Preferably, any external input such as user interaction is converted to OCP and/or ODP before it is presented to the executive. There is no need for headers in a bitstream delivered over a network. However, headers are required when storing an MPEG4 presentation in a file.

Figs. 2a and 2b illustrate the structure of composition and data packets in further detail. Relevant features are as follows:

Object ID is composed of object type and object number. The default length of the Object ID is 2 bytes, including ten bits for the object number and 6 for the object type (e.g. text, graphics, MPEG2 VOP, compound object). An extensible code is used to accommodate more than 1023 objects or more than 31 object types. The following convention will be adhered to: a value of 0b111111 in the first six bits of the Object ID corresponds to 31 plus the value of the byte immediately following the ObjectID; a value of 0b11.1111.1111 in the least significant 10 bits of the Object ID corresponds to 1023 plus the value of the two bytes immediately following the Object ID (without counting the object type extension bytes, if present). The following object types

are defined:

Composition Objects (16-bit object IDs)

0X0000	scene configuration object
0X0001	node hierarchy specification
0X0002	stream-node association
0X0003	node/scene update
0X0004	compound object

Object Data (object type, 6 most significant bits)

0b00.0010	text
0b00.0011	MPEG2 VOP (rectangular VOP)

Persistent Objects (PO) are objects that should be saved at the decoder for use at a later time. An expiration time stamp (ETS) gives the life of a PO in milliseconds. A PO is not available to the decoder after ETS runs out. When a PO is to be used at a later time in a scene, only the corresponding composition information needs to be sent to the AV terminal.

Decoding Time Stamp (DTS) indicates the time an object (access unit) should be decoded by the decoder.

Presentation Time Stamp (PTS) indicates the time an object (access unit) should be presented by the decoder.

Lifetime Time Stamp (LTS) gives the duration (in milliseconds) an object should be displayed in a scene. LTS is implicit in some cases, e.g. in a video sequence where a frame is displayed for 1/frame-rate or until the next frame is available, whichever is larger. An explicit LTS is used when displaying graphics and text. An AV object should be decoded only once for use during its life time.

Expiration Time Stamp (ETS) is specified to support the notion of object persistence. An object, after it is presented, is saved at the decoder (cache) until a time given by ETS. Such an object can be used multiple times before ETS runs out. A PO with an expired ETS is no

longer available to the decoder.

Object Time Base (OTB) defines the notion of time of a given AV object encoder. Different objects may belong to different time bases. The AV terminal adapts these time bases to the local one, as specified in the MSDL VM.

Object Clock Reference (OCR) can be used if necessary to convey the speed of the OTB to the decoder. By this mechanism, OTBs can be recovered/adapted at the AV terminal.

Composition Parameters are used to compose a scene (place an object in a scene). These include displacement from the upper left comer of the presentation frame, rotation angles, zooming factors, etc.

Priority indicates the priority of an object for transmission, decoding, and display. MPEG4 supports 32 levels of priority. Lower numbers indicate higher priorities.

Persistence Indicator (PI) indicates whether an object is persistent.

Continuation Indicator (CI) indicates the end of an object in the current packet (or continuation).

Object Grouping facilitates operations to be applied to a set of objects with a single operation. Such a feature can be used to minimize the amount of composition information sent, as well as to support hierarchical scene composition based on independent sub-scenes. The composer manipulates the component objects as a group. The structure of a compound composition packet (CCP) is shown in Fig. 2c.

Bitstream Structure includes object composition packets for describing the composition and controlling the presentation of those packets, and object data packets that contain the data for the objects. A scene is composed by a set of composition packets. The

bitstream supports representation of scenes as a hierarchy by using compound composition objects (CCP), also known as node hierarchy. A CCP allows combining composition objects to create complex audio-visual scenes.

Object-Data Packet:

ObjectID - min (default) 10 bits

CI and PI could be combined:

00 - begin non-persistent

01 - begin persistent

10 - continuation

11 - end of object

Priority: 5 bits, present only if CI/PI is 0b00 or 0b01

ETS: 30 bits, present if CI/Pl is 0b01

For prediction-based video coding, VOP_type is indicated by two bits (00 (I), 01 (P), 10 (B), 11 (PB)), facilitating editing.

```
Object_data_packet{
    ObjectID                    16 bits + any extensions;
    CIPI                        2 bits
    if  (CIPI <= 1){
    Priority                    5 bits
    if (object type is MPEG VOP)
                    (any prediction based compression)
        VOP_type                    2 bits
    }
    if (CIPI == 1)
    ETS                         28  bits
    ObjectData
}
```

Object Composition Packet

```
Object_composition_packet{
    ObjectID                    16 bits + any extensions
    OCR_Flag                    1 bit
```

```
        Display_Timers_Flag          1 bit
        DTS                          30  bits
        if (OCR_Flag)
             OCR                     30  bits
        if (Display_Timers_Flag){
             PTS                     30  bits
             LTS                     28  bits
        }
        Composition_parameters;
}
```

Composition_Parameters are defined in section 2 of MSDL Verification Model, MPEG N1483, Systems Working Draft V2.0, the disclosure of which is incorporated herein by reference.

```
Composition_parameters(
        visibility                        1 bit
             composition_order            5 bits
             number_of_motion_sets        2 bits
             x_delta_0                    12 bits
             y_delta_0                    12 bits
             for  (i = 1;  i <= number_of_motion_sets; i++){
                  x_delta_i               12 bits
                  y_delta_i               12 bits
             }
}
```

Compound_Composition_Packet

```
Compound_composition_packet{
        ObjectID                          16 bits
        PTS                               30 bits
        LTS                               28 bits
        Composition_parameters
        ObjectCount                       8 bits
        for   (i = 0; i < ObjectCount; i++){
                  Object_composition_packet;
```

```
        }
    }
```

Scene Configuration Packet (SCP) is used to change reference scene width, height, to flush the buffer, and other configuration functions. The object type for SCPs is 0b00.0000. This allows for 1024 different configuration packets. The object number 0b00.0000.0000 (object ID 0X0000) is defined for use with flushing the terminal buffers.

Composition Control for Buffer Management (Object ID 0x0000)

AV terminal buffers are flushed using Flush_Cache and Scene_Update flags. When using hierarchical scene structure, the current scene graph is flushed and the terminal loads the new scene from the bitstream. Use of flags allows for saving the current scene structure instead of flushing it. These flags are used to update the reference scene width and height whenever a new scene begins. If the Flush_Cache_Flag is set, the cache is flushed, removing the objects (if any). If Scene_Update_Flag is set, there are two possibilities: (i) Flush_Cache-Flag is set, implying that the objects in the cache will no longer be used; (ii) Flush_Cache_Flag is not set, the new scene being introduced (an editing action on the bitstream) splices the current scene and the objects in the scene will be used after the end of the new scene. The ETS of the objects, if any, will be frozen for the duration of the new scene introduced. The beginning of the next scene is indicated by another scene configuration packet.

```
Scene_configuration_packet{
    ObjectID                    16 bits (OXOOOO)
        Flush_Cache_Flag        1 bit
```

```
Scene_Update_Flag          1 bit
if (Scene_Update_Flag){
     ref_scene_width       12 bits
     ref_scene_height      12 bits
}
}
```

Composition Control for Scene Description (Object ID 0x0001)

A hierarchy of nodes is defined, describing a scene. The scene configuration packets can also be used to define a scene hierarchy that allows for a description of scenes as a hierarchy of AV objects. Each node in such a graph is a grouping of nodes that groups the leaves and/or other nodes of the graph into a compound AV object. Each node (leaf) has a unique ID followed by its parameters as shown in Fig. 3.

Composition Control for Stream-Node Mapping (Object ID 0x0002)

As illustrated by Fig. 4, table entries associate the elementary object streams in the logical channels to the nodes in a hierarchical scene. The stream IDs are unique, but not the node IDs. This implies that more than one stream can be associated with the same node.

Composition Control for Scene Updates (Object ID 0x0003)

Fig. 5 illustrates updating of the nodes in the scene hierarchy, by modifying the specific parameters of the node. The graph itself can be updated by adding/deleting the nodes in the graph. The update type in the packet indicates the type of update to be performed on the graph.

Architectural Embodiment

The embodiment described below includes an

object-based AV bitstream and a terminal architecture. The bitstream design specifies, in a binary format, how AV objects are represented and how they are to be composed. The AV terminal structure specifies how to decode and display the objects in the binary bitstream.

AV Terminal Architecture

Further to Fig. 1 and with specific reference to Fig. 6, the input to the de-multiplexer 1 is an object-based bitstream such as an MPEG-4 bitstream, consisting of AV objects and their composition information multiplexed into logical channels (LC). The composition of objects in a scene can be specified as a collection of objects with independent composition specification, or as a hierarchical scene graph. The composition and control information is included in LC0. The control information includes control commands for updating scene graphs, reset decoder buffers etc. Logical channels 1 and above contain object date. The system includes a controller (or "executive") 2 which controls the operation of the AV terminal.

The terminal further includes input buffers 3, AV object decoders 4, buffers 4' for decoded data, a composer 5, a display 6, and an object cache 7. The input bitstream may be read from a network connection or from a local storage device such as a DVD, CD-ROM or computer hard disk. LC0 containing the composition information is fed to the controller. The DMUX stores the objects in LC1 and above at the location in the buffer specified by the controller. In the case of network delivery, the encoder and the stream server cooperate to ensure that the input object buffers neither overflow nor underflow. The encoded data objects are stored in the input data buffers until read by the decoders at their decoding time, typically given by an

associated decoding timestamp. Before writing a data object to the buffer, the DMUX removes the timestamps and other headers from the object data packet and passes them to the controller for signaling of the appropriate decoders and input buffers. The decoders, when signaled by the controller, decode the data in the input buffers and store them in the decoder output buffers. The AV terminal also handles external input such as user interaction.

In the object cache 7, objects are stored for use beyond their initial presentation time. Such objects remain in the cache even if the associated node is deleted from the scene graph, but are removed only upon the expiration of an associated time interval called the expiration time stamp. This feature can be used in presentations where an object is used repeatedly over a session. The composition associated with such objects can be updated with appropriate update messages. For example, the logo of the broadcasting station can be downloaded at the beginning of the presentation and the same copy can be used for repeated display throughout a session. Subsequent composition updates can change the position of the logo on the display. Objects that are reused beyond their first presentation time may be called persistent objects.

<u>System Controller(SC)</u>

The system controller controls decoding and playback of bitstreams on the AV terminal. At startup, from user interaction or by looking for a session at default network address, the SC first initializes the DMUX to read from a local storage device or a network port. The control logic is loaded into the program RAM at the time of initialization. The instruction decoder reads the instructions from the program and executes

them. Execution may involve reading the data from the input buffers (composition or external data), initializing the object timers, loading or updating the object tables to the data RAM, loading object timers, or control signaling.

Fig. 7 shows the system controller in further detail. The DMUX reads the input bitstream and feeds the composition data on LC0 to the controller. The composition data begins with the description of the first scene in the AV presentation. This scene can be described as a hierarchical collection of objects using compound composition packets, or as a collection of independent object composition packets. A table that associates the elementary streams with the nodes in the scene description immediately follows the scene description. The controller loads the object IDs (stream IDs) into object list and render list which are maintained in the data RAM. The render list contains the list of objects that are to be rendered on the display device. An object that is disenabled by user interaction is removed from the render list. A node delete command that is sent via a composition control packet causes the deletion of the corresponding object IDs from the object list. The node hierarchy is also maintained in the data RAM and updated whenever a composition update is received.

The composition decoder reads data from the composition and external data buffer and converts them into a format understood by the instruction decoder. The external input includes user interaction to select objects, disenable and enable objects and certain predefined operations on the objects. During the execution of the program, two lists are formed in the data RAM. The object list, containing a list of objects

(object IDs) currently handled by the decoders and a render list, containing the list of active objects in the scene. These lists are updated dynamically as the composition information is received. For example, if a user chooses to hide an object by passing a command via the external input, the object is removed from the render list until specified by the user. This is also how external input is handled by the system. Whenever there is some external interaction, the composition decoder reads the external data buffer and performs the requested operation.

The SC also maintains timing for each AV object to signal the decoders and decoder buffers of decoding and presentation time. The timing information for the AV objects is specified in terms of its time-base. The terminal uses the system clock to convert an object's time base into system time. For objects that do not need decoding, only presentation timers are necessary. These timers are loaded with the decoding and presentation timestamps for that AV object. The controller obtains the timestamps from the DMUX for each object. When a decoding timer for an object runs out, the appropriate decoder is signaled to read data from the input buffers and to start the decoding process. When a presentation timer runs out, the decoded data for that object is transferred to the frame buffer for display. A dual buffer approach could be used to allow writing to a frame buffer while the contents of the second buffer are displayed on the monitor. The instruction decoder can also reset the DMUX or input buffers by signaling a reset, which initializes them to the default state.

Information Flow in the Controller

Fig. 8 shows the flow of information in the controller. To keep the figure simple, the operations

performed by the instruction decoder are shown in groups. The three groups respectively concern object property modifications, object timing, and signaling.

Object Property Modifications

These operations manipulate the object IDs, also called elementary stream IDs. When a scene is initially loaded, a scene graph is formed with the object IDs of the objects in the scene. The controller also forms and maintains a list of objects in the scene (object list) and active objects in the object from the render list. Other operations set and update object properties such as composition parameters when the terminal receives a composition packet.

Object Timing

This group of operations deals with managing object timers for synchronization, presentation, and decoding. An object's timestamp specified in terms of its object time base is converted into system time and the presentation and decoding time of that object are set. These operations also set and reset expiration timestamps for persistent objects.

Signaling

Signaling operations control the over-all operation of the terminal. Various components of the terminal are set, reset and operated by controller signaling. The controller checks the decoding and presentation times of the objects in the render list and signals the decoders and presentation frame buffers accordingly. It also initializes the DEMUX for reading from a network or a local storage device. At the instigation of the controller, decoders read the data from the input buffers and pass the decoded data to decoder output buffers. The decoded data is moved to the presentation device when signaled by the controller.